JBoss Community

Red Hat Research Away Day

Introduction

- SAVARA has been established to provide methodology and tool support, for building large scale Enterprise Architectures, based on a concept called *Testable Architecture*

- A *Testable Architecture* can be defined as:
  "... one where artifacts, at any stage within the software development lifecycle, can be verified against artifacts in preceding stages. This leads to deployed systems that can be shown to meet the original business requirements."

Testable Architecture – why do we need it?

- Historically business requirements, architectures and designs have not been documented in a manner that can support verification across all phases
- Emphasis has been on unit and integration testing to determine if individual services and complete system meets the original business requirements
- However – what ensures that these unit and integration tests correctly encode the business requirements?
- How do we ensure that the architecture and design documents continue to reflect the implementation as the system evolves?

Testable Architecture – why do we need it? (2)

- Testable architecture gives:
  - Accuracy: Higher level of confidence that business requirements are implemented by deployed system
  - Efficiency: Improves communication between project members, to increase effectiveness of software development, especially when geographically distributed
  - Quality: Design time process governance enables earliest possible detection of misalignment to requirements thus reducing cost of errors
  - Fidelity: Runtime process governance support for continuous validation of production system

Testable Architecture – where did it come from?

- W3C Choreography Working Group produced WS-CDL
  - Goal to have formal underpinnings based on Pi Calculus
  - First choreography notation based on the “global model” approach, inspired by the work of Lucian Wischik on fusions
  - First collaboration with Prof. Robin Milner, Dr Kohei Honda and Dr Nobuko Yoshida
- pi4soa open source project
  - Implementation of WS-CDL
  - Introduced scenarios for defining interaction based use cases
  - Verification of scenarios against CDL was first step towards a 'testable architecture' approach

Testable Architecture – where did it come from?

- Red Hat Project Overlord
  - Process Governance (design and run-time)
- Scribble 1
  - Aim was to provide a simple text based notation for describing interaction based behaviour that was more natural to users than pi-calculus notation
  - Issue was that theoretical research around the global model was in its infancy
- SAVARA 1
  - Enterprise architecture/solutions tool suite incorporating process governance
  - Mainly based on pi4soa tools, although Scribble 1 used for initial experimentation with conformance checking

Savara 1 Movies

- Movies of Savara 1 can be found here:
  - http://www.jboss.org/savara/documentation/movies

- These show different phases of the Testable Architecture methodology, including static and dynamic governance.

Case Study 1: Global Insurance Company (1)

- How did Cognizant do it?
  - Two streams one with and without testable architecture but the same problem set.

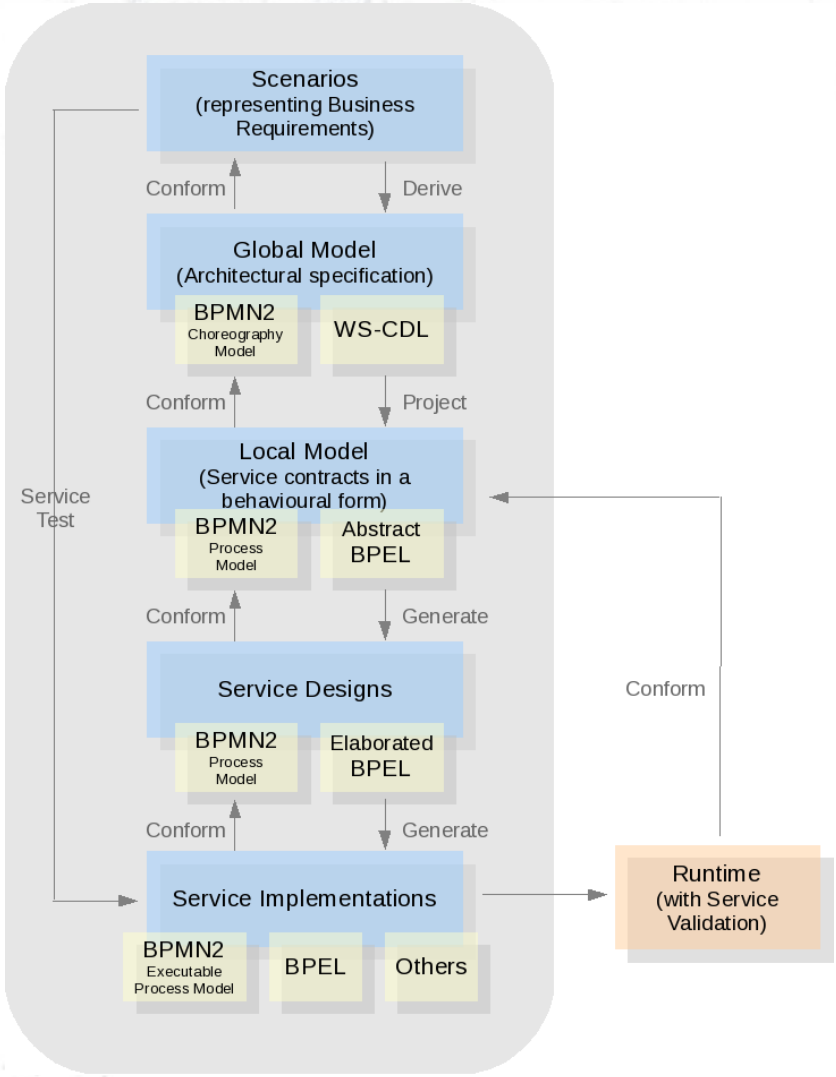
- Why did Cognizant do it?
  - Because they wanted to align requirements from BA's to solutions in a testable way prior to coding in order to reduce the risk of mis-delivery, reduce design-time errors, reduce the cost and time of delivery and increase the quality of the delivery.

Case Study 1: Global Insurance Company (2)

- What was achieved?
  - They aligned solutions to requirements in a testable way and as a result:
    - Found and removed errors at design time with a saving of more than 20% over the entire SDLC
    - Reduced time to gather requirements and define the architecture by 80%
  - And this was a small project.

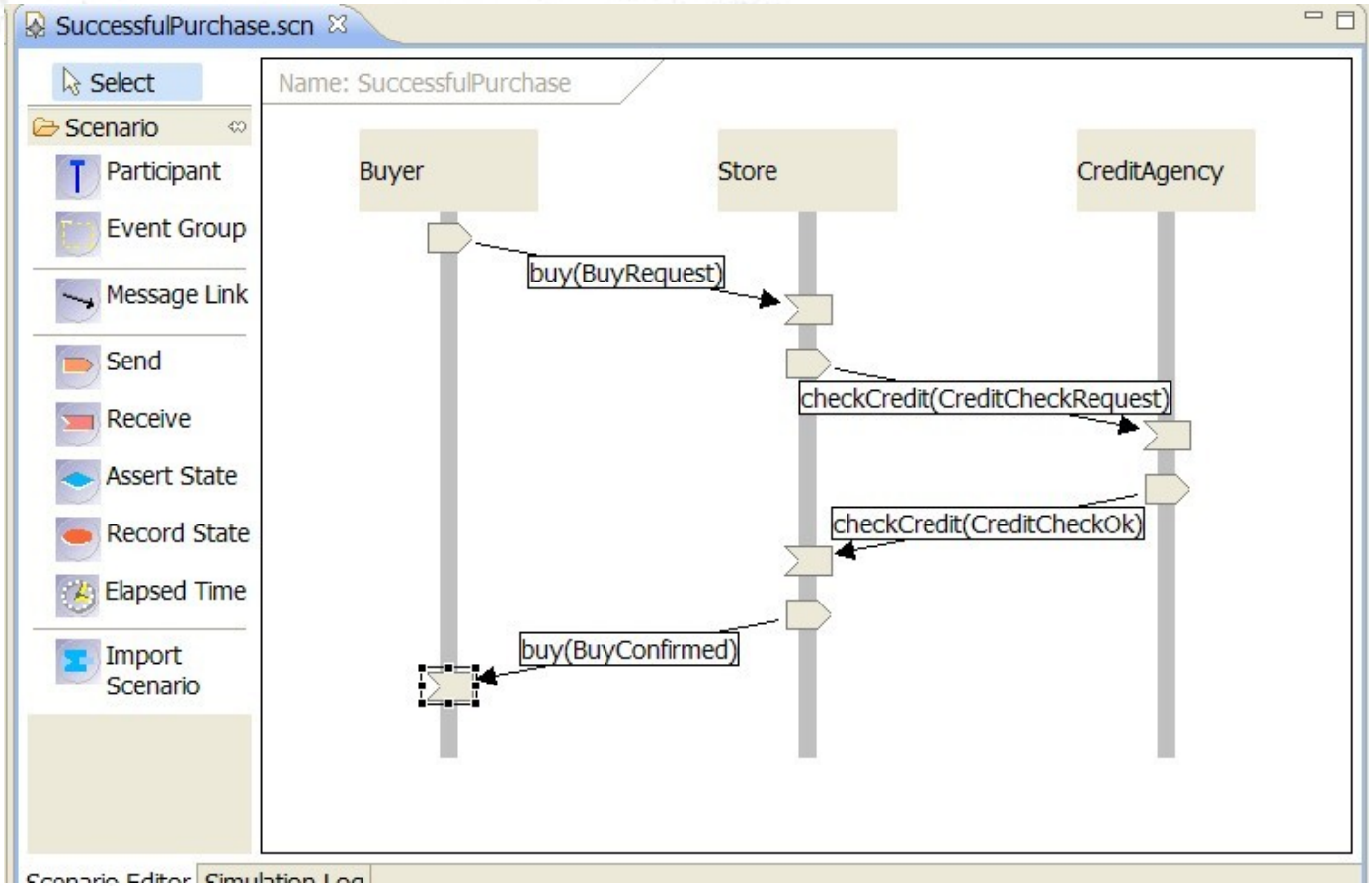
Case Study 2: Large Retail Bank

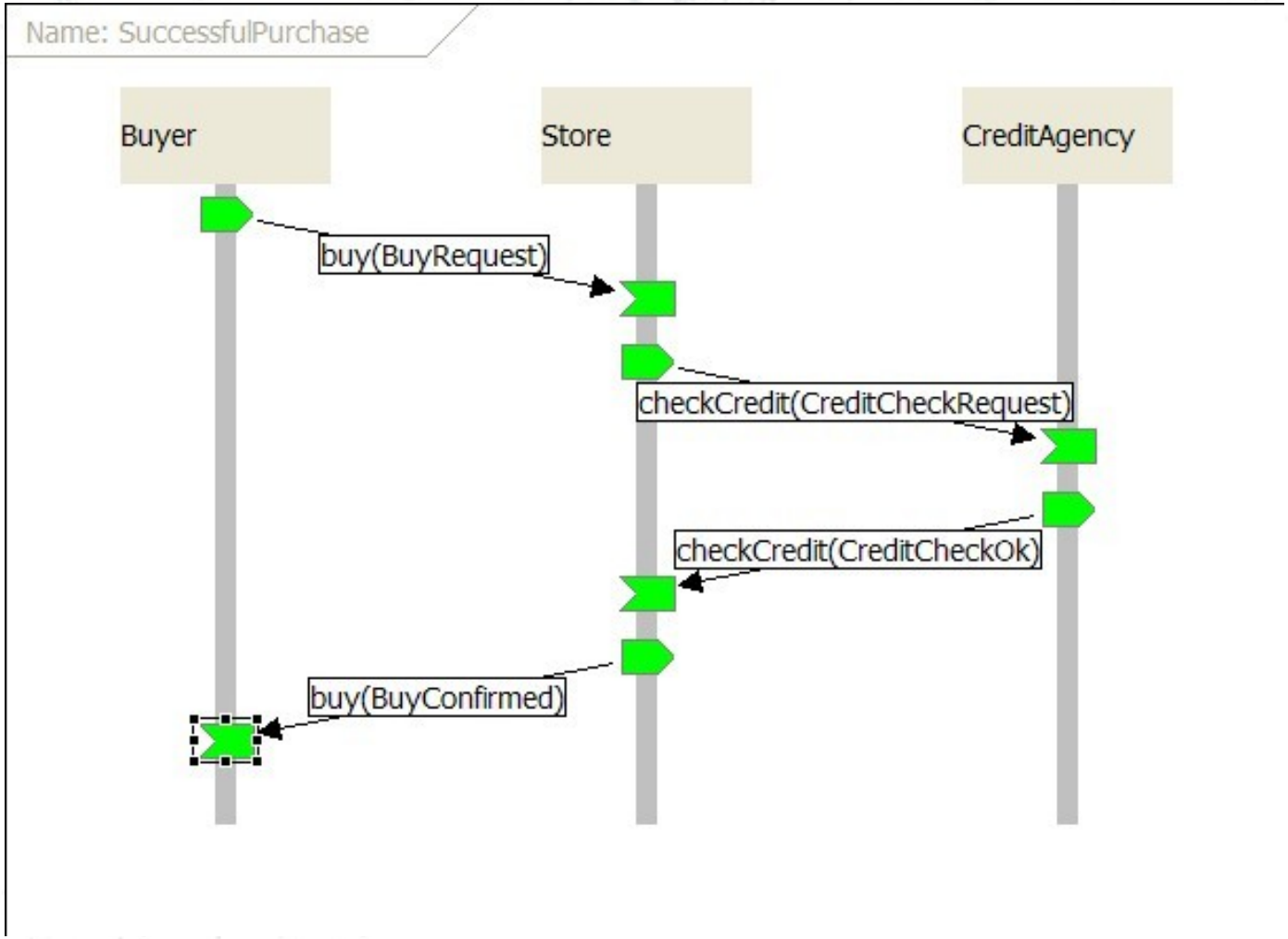
- One medium defect took 47 hours to determine where the problem was using 3 teams of 5 people.
- When the same input data (log files) was supplied to Systemic Defect Profiler (incorporating savara) it took 2 minutes to come to the same conclusion.
- Net result is an average drop in the cost of quality of 51%.

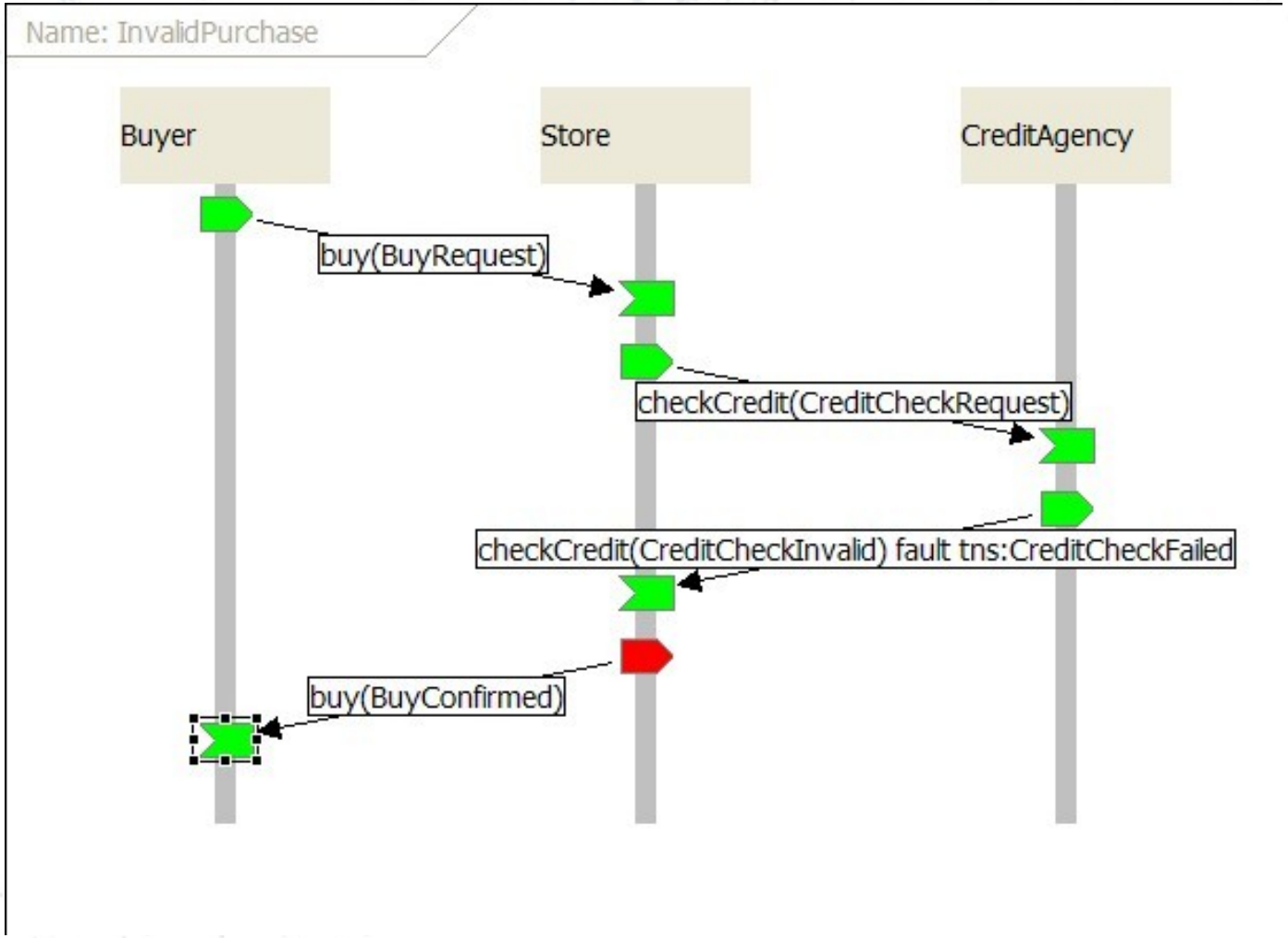

Scenarios
(representing Business Requirements)
Conform
Derive
Global Model
(Architectural specification)
BPMN2
Choreography Model
WS-CDL
Conform
Project
Local Model
(Service contracts in a behavioural form)
BPMN2
Process Model
Abstract BPEL
Conform
Generate
Service Designs
BPMN2
Process Model
Elaborated BPEL
Conform
Generate
Service Implementations
BPMN2
Executable Process Model
BPEL
Others
Service Test
Conform
Runtime
(with Service Validation)

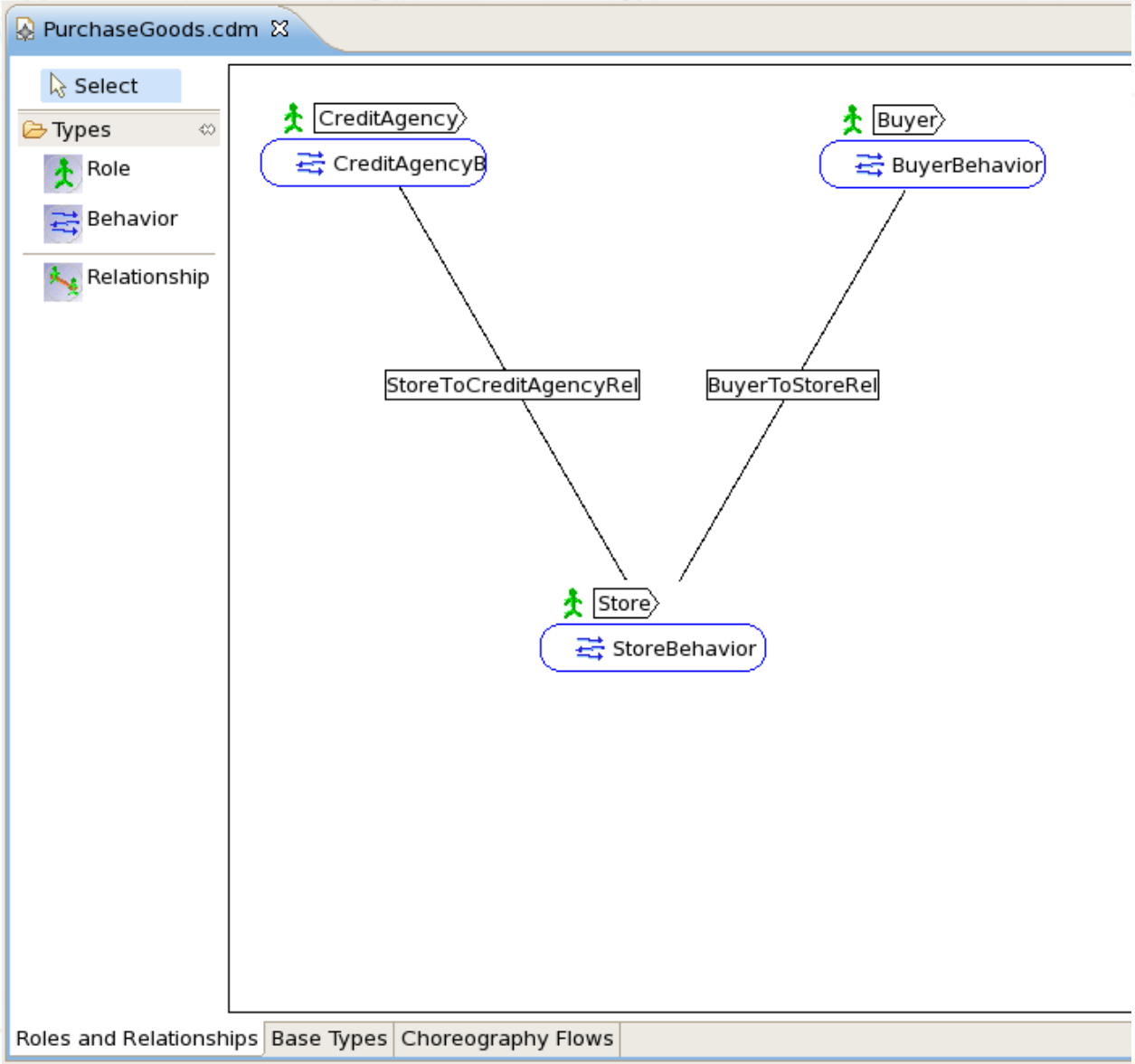
Step 1: Business Analysis and Architecture

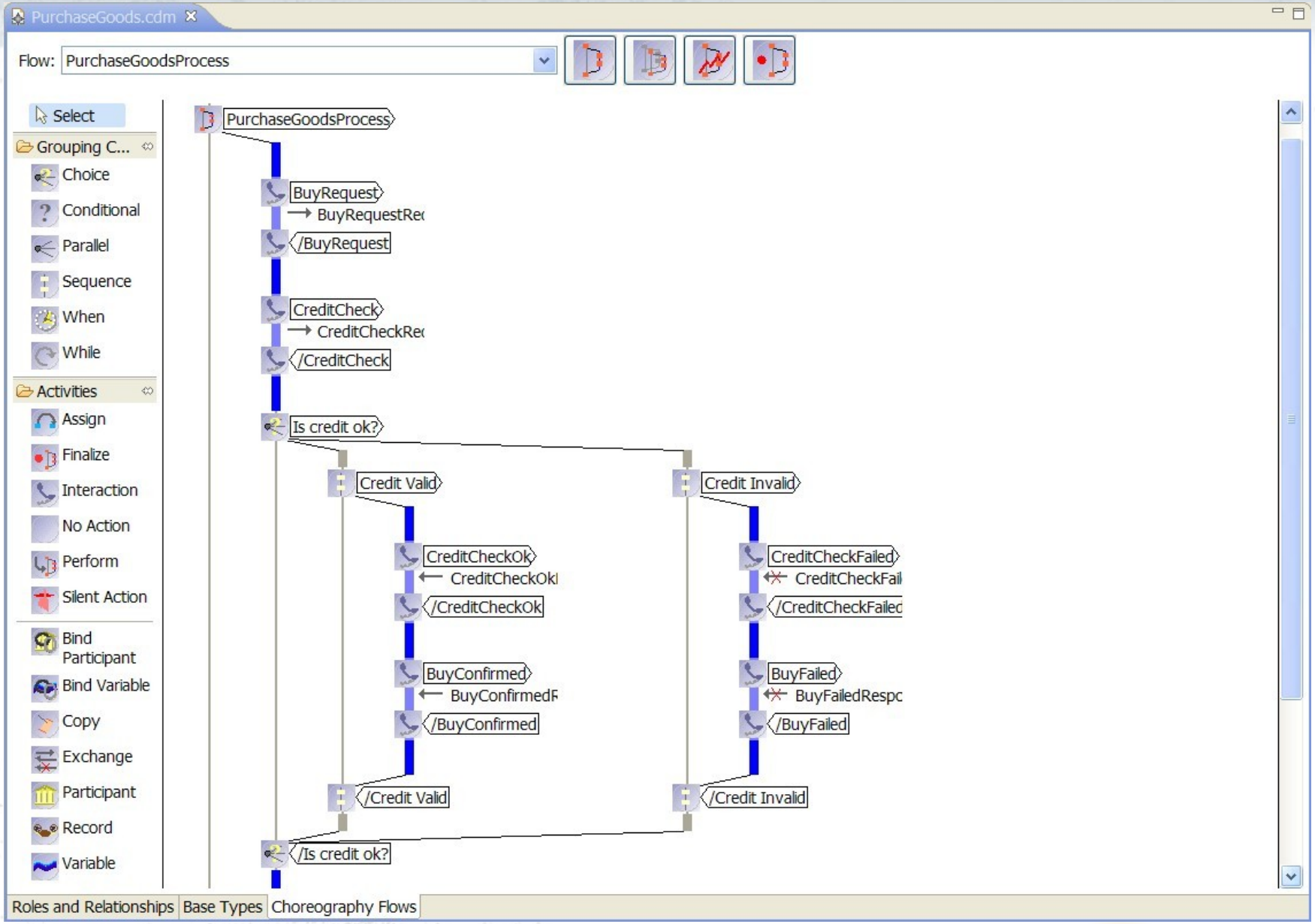
- Requirements
  - Defined as communication based scenarios (or sequence diagrams) with example messages
- Global Model or Choreography
  - Represents service neutral perspective of interactions between distributed parties
- Information Model
  - Schema derived from relevant example messages associated with scenarios
- Optional Outline Deployment Model


SuccessfulPurchase.scn
Select
Scenario
Participant
Event Group
Message Link
Send
Receive
Assert State
Record State
Elapsed Time
Import Scenario
Name: SuccessfulPurchase
Buyer
Store
CreditAgency
buy(BuyRequest)
checkCredit(CreditCheckRequest)
checkCredit(CreditCheckOk)
buy(BuyConfirmed)


Name: SuccessfulPurchase
Buyer
Store
CreditAgency
buy(BuyRequest)
checkCredit(CreditCheckRequest)
checkCredit(CreditCheckOk)
buy(BuyConfirmed)


Name: InvalidPurchase
Buyer
Store
CreditAgency
buy(BuyRequest)
checkCredit(CreditCheckRequest)
checkCredit(CreditCheckInvalid) fault tns:CreditCheckFailed
buy(BuyConfirmed)


PurchaseGoods.cdm
Select
Types
Role
Behavior
Relationship
CreditAgency
CreditAgencyB
Buyer
BuyerBehavior
StoreToCreditAgencyRel
BuyerToStoreRel
Store
StoreBehavior
Roles and Relationships
Base Types
Choreography Flows

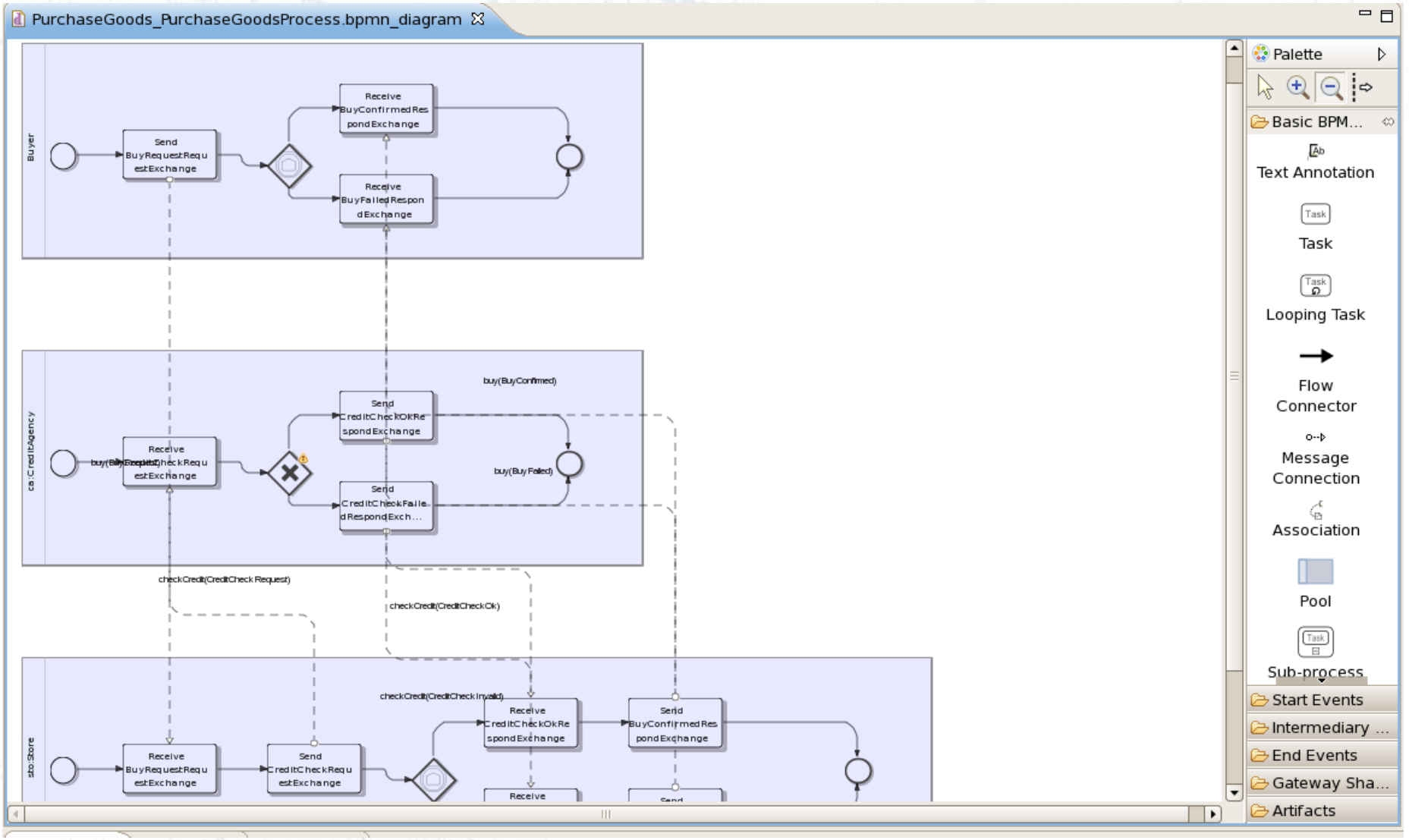

PurchaseGoods.cdm
Flow: PurchaseGoodsProcess
Select
Grouping C...
Choice
Conditional
Parallel
Sequence
When
While
Activities
Assign
Finalize
Interaction
No Action
Perform
Silent Action
Bind Participant
Bind Variable
Copy
Exchange
Participant
Record
Variable
PurchaseGoodsProcess
BuyRequest
BuyRequestRe
/BuyRequest
CreditCheck
CreditCheckRe
/CreditCheck
Is credit ok?
Credit Valid
Credit Invalid
CreditCheckOk
CreditCheckOkI
/CreditCheckOk
CreditCheckFailed
CreditCheckFail
/CreditCheckFailed
BuyConfirmed
BuyConfirmedF
/BuyConfirmed
BuyFailed
BuyFailedRespc
/BuyFailed
/Credit Valid
/Credit Invalid
/Is credit ok?
Roles and Relationships
Base Types
Choreography Flows

Step 2: Service Analysis

- Identify Service Candidates
- Local Model, representing an abstract behavioural contract for the service component
  - Does not necessarily need to be persisted
  - Could be stored in SOA Repository and used for service lookup based on behavioural compatibility
- Service Level Agreements
  - Policies governing contractual obligations in terms of properties such as availability and performance
  - Can be associated with the service contract as defined by the Local Model

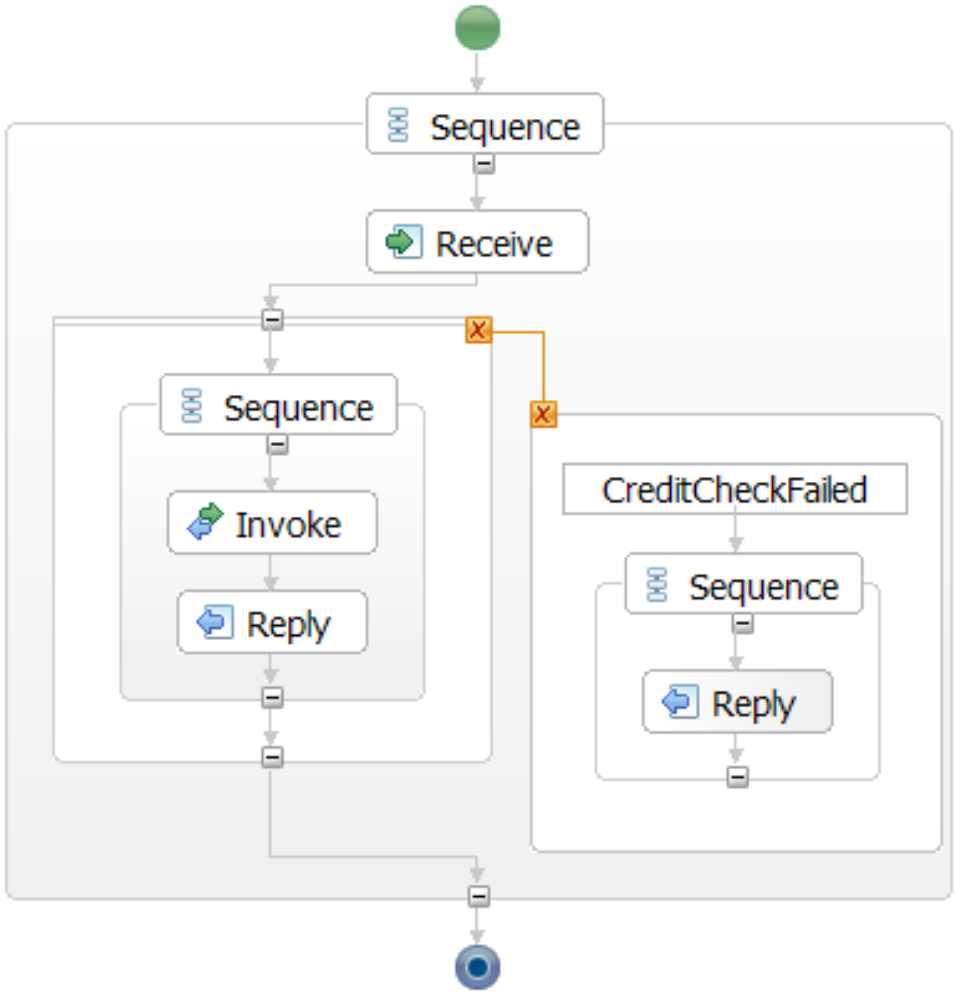
Step 3: Service Development (1)

- Testable architecture approach facilitates distributed development of services
  - each service is precisely defined by its abstract service contract and SLAs
  - isolated conformance checking and testing against scenarios reduce issues during integration testing
- Service Design
  - Elaborated Local Model, can be conformance checked against persisted local, or projected global, model


PurchaseGoods_PurchaseGoodsProcess.bpmn_diagram
Buyer
Send BuyRequestRequestExchange
Receive BuyConfirmedRespondExchange
Receive BuyFailedRespondExchange
ca:CreditAgency
Receive CreditCheckRequestExchange
Send CreditCheckOkRespondExchange
Send CreditCheckFailedRespondExch...
buy(BuyConfirmed)
buy(BuyFailed)
checkCredit(CreditCheckRequest)
checkCredit(CreditCheckOk)
checkCredit(CreditCheckInvalid)
sto:Store
Receive BuyRequestRequestExchange
Send CreditCheckRequestExchange
Receive CreditCheckOkRespondExchange
Send BuyConfirmedRespondExchange
Palette
Basic BPM...
Text Annotation
Task
Looping Task
Flow Connector
Message Connection
Association
Pool
Sub-process
Start Events
Intermediary ...
End Events
Gateway Sha...
Artifacts

Step 3: Service Development (2)

- Data Model Design
  - Database can be viewed as services, where interactions in a scenario represent queries to the database
  - Enables database to be verified against scenarios
- Service Implementation
  - Generation and conformance checking of implementation against Service Design or Local Model
  - Targets include BPEL, executable BPMN2, SCA, Switchyard

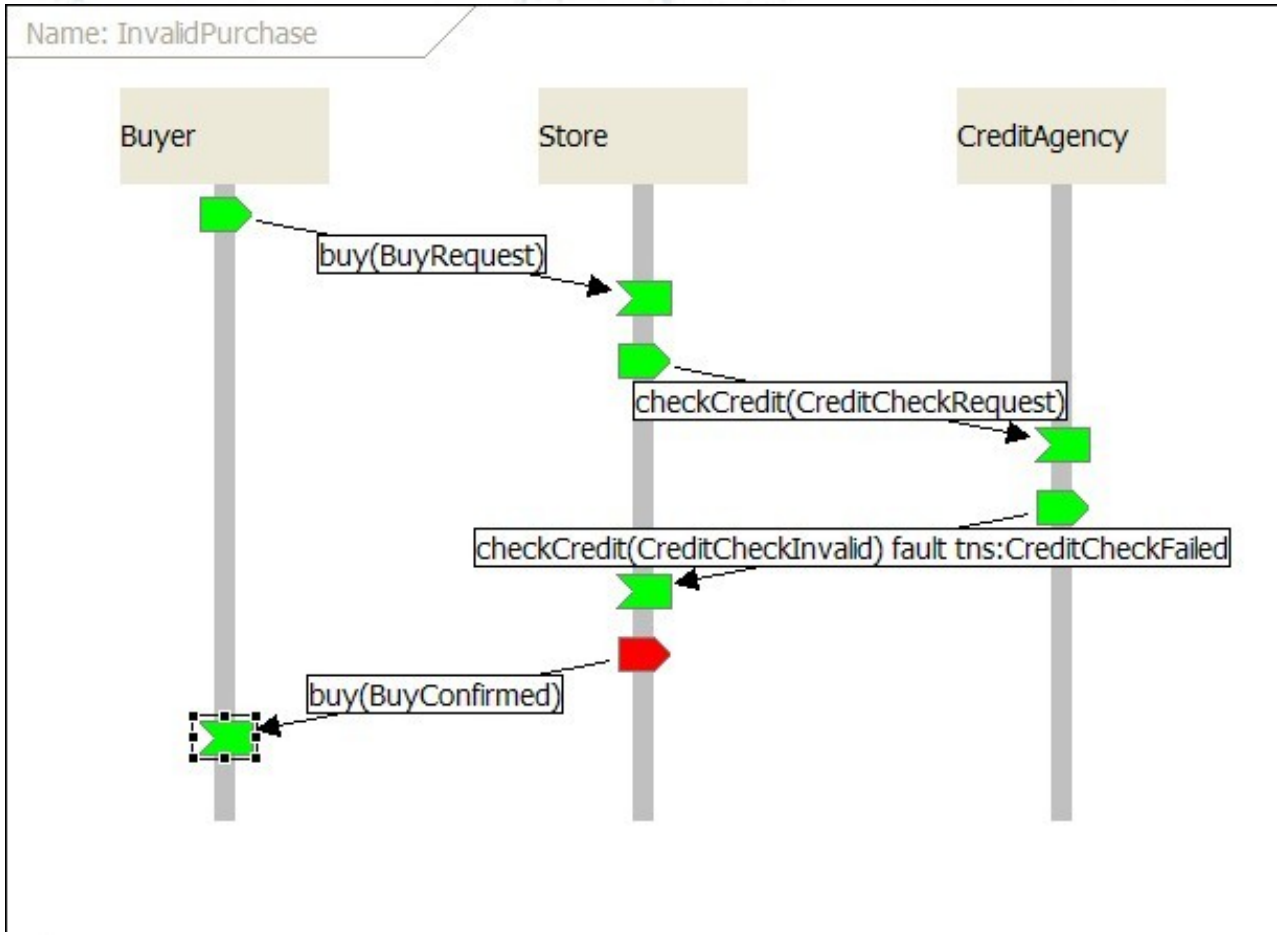

Sequence
Receive
Sequence
Invoke
Reply
CreditCheckFailed
Sequence
Reply

Step 3: Service Development (3)

- Optional Detailed Deployment Model
  - Will elaborate the outline deployment model
  - Previous service generation to specific implementation languages can also be used to annotate relevant aspects of the deployment model (if available)
  - Deployment model can provide runtime technology specific information that could aid deployment of service implementations to the relevant service containers

Step 4: Testing

- Component Unit Testing
  - Enables 'components' within a global model, such as services, databases, human interfaces, etc. to be tested in the context of the original requirements (i.e. scenarios)
  - Test inputs obtained from scenario example messages
  - Test outputs from component under test will be compared against expected scenario example message
- System Integration Testing
  - Using runtime monitoring to check complete system


Name: InvalidPurchase
Buyer
Store
CreditAgency
buy(BuyRequest)
checkCredit(CreditCheckRequest)
checkCredit(CreditCheckInvalid) fault tns:CreditCheckFailed
buy(BuyConfirmed)

Step 5: Documentation

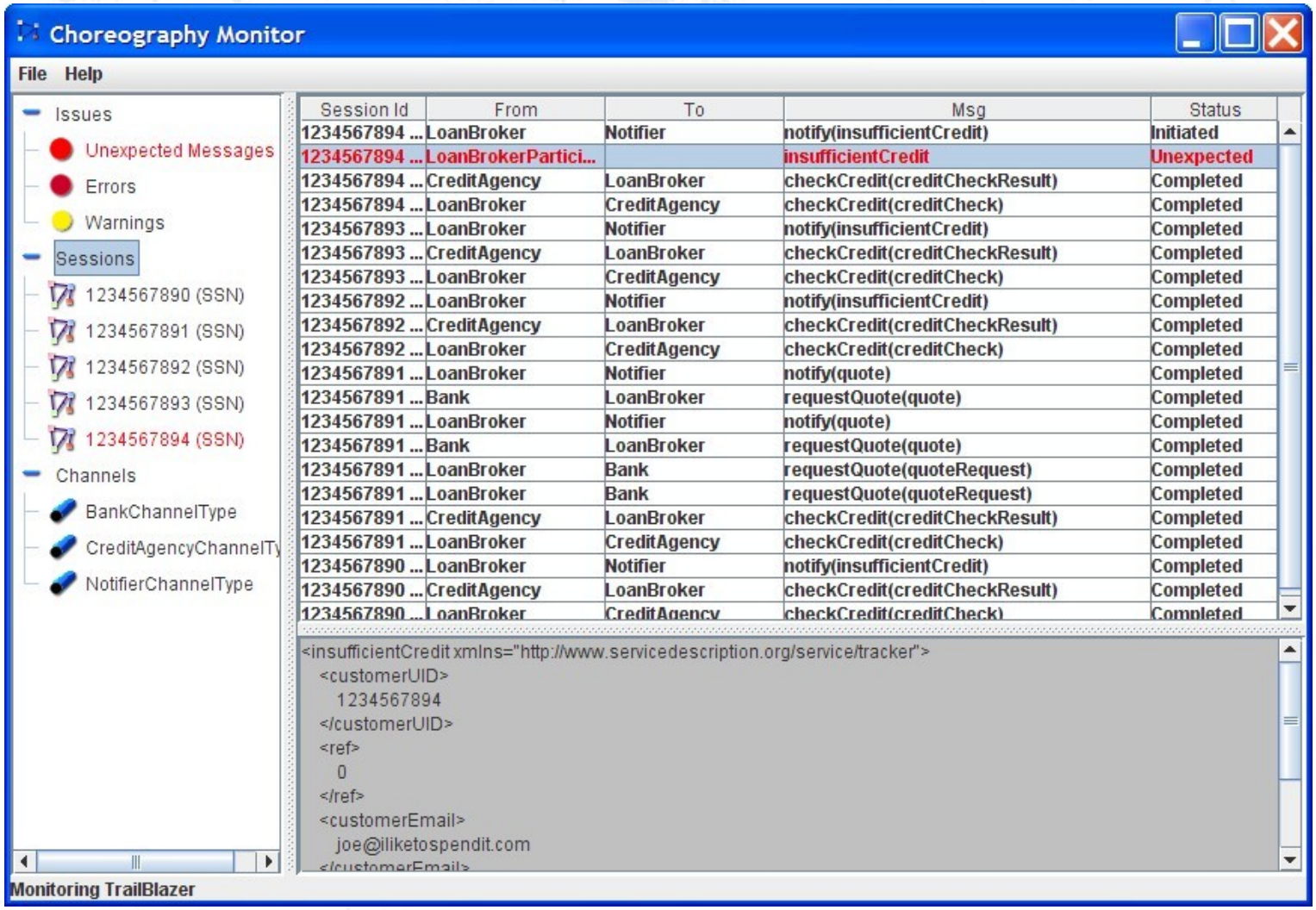
- Testable architecture enables system to be defined, from requirements to implementation, in a verifable manner
- No unstructured documentation of requirements or design is required, which tend to become out of date very quickly
- However documentation produced from verifiable artifacts will still be required, to promote understanding of the system, and support sign-off of relevant phases of the project

Step 6: Deployment

- If a deployment model has been defined, then it can be used to guide the deployment of service implementations to the relevant service containers
- Extensibility will be required to support a range of deployment environments

Step 7: Runtime Monitoring

- Monitor the runtime execution of the system against the Global or Local Models
  - Can be used as continuous verification
  - Useful where some service components not developed using Testable Architecture (i.e. 3rd party services), or where the implementation language used could not be checked for conformance against a suitable model
- Assertions, constraints and SLAs, defined in the Global or Local Models, can be monitored with violations being reported



JBoss Community

SAVARA

Testable Architecture – where is it going?

- Scribble 2
  - Global model (or session type) research has progressed a lot since first version of Scribble.
  - Global model is lock free by design, as long as linearity is preserved, and all roles are projectable.
  - Scribble related research now being carried out at various universities including Queen Mary, Imperial and Leicester.
  - Guiding principle now is to define scribble constructs that have a theoretical grounding – almost there.
  - Official version 1 of the notation should be available soon.

Testable Architecture – where is it going?

- SAVARA 2
  - No longer WS-CDL and Eclipse centric.
  - Modular OSGi core based around Scribble 2 as the canonical representation
  - Enables use with Eclipse, JBoss AS container, embedded, etc
  - Focusing more around BPMN2, due to support for choreography and service/process (endpoint) models
  - Activity monitoring framework and web apps being developed
  - SCA Java, BPEL and BPMN2 process model generation targets
  - Centred around SOA Repository for artifacts, representing the dependencies that can be used as the basis for validation

Ocean Observatories Initiative

- What is OOI?
  - environmental observatory covering a diversity of oceanic environments, ranging from the coastal to the deep ocean
  - a comprehensive cyberinfrastructure whose design is based on loosely coupled distributed services
  - elements are expected to reside throughout the OOI observatories, from seafloor instruments to deep sea moorings to shore facilities to computing and archiving infrastructure
  - infrastructure expected to have an operating life of 30 years

Ocean Observatories Initiative (2)

- How is SAVARA involved?
  - In such a highly distributed infrastructure, self governance is important
  - Parts of infrastructure will be owned and operated by different organizations, so internal policies (e.g. usage of components) need to be managed and policed
  - Working with Queen Mary, Imperial and Leicester Universities to build small lightweight efficient protocol+assertion monitoring solution

Research – scribble language (1)

```
protocol PurchaseGoods (role Buyer) {
        Buyer introduces Store;
        Store introduces CreditAgency;

        buy(BuyRequest) from Buyer to Store;
        checkCredit(CreditCheckRequest) from Store to CreditAgency;

        choice at CreditAgency {
                checkCredit(CreditRating) from CreditAgency to Store;

                choice at Store {
                        Store introduces Logistics;

                        deliver(DeliveryRequest) from Store to Logistics;
                        deliver(DeliveryConfirmed) from Logistics to Store;
                        buy(BuyConfirmed) from Store to Buyer;
                } or {
                        buy(BuyFailed) from Store to Buyer;
                }
        } or {
                checkCredit(CustomerUnknown) from CreditAgency to Store;
                buy(AccountNotFound) from Store to Buyer;
        }
}
```

Research – scribble language (2)

- Concurrency

```
parallel {
  M1 from A to B;
} and {
  M2 from A to C;
}
```

- Repetition

```
repeat at A {
  M1 from A to B;
}
```

- Recursion

```
txn:
  choice at A {
        M1 from A to B;
        txn;
  } or {
        M2 from A to B;
  }
```

Research – scribble language (3)

- Calling Nested and External Protocols

  run OtherProtocol (A, B);

- Global escape – interrupt mechanism

```
Order from Customer to Supplier;
do {
  Confirm from Supplier to Customer;
} interrupt {
  Cancel from Customer to Supplier;
}
```

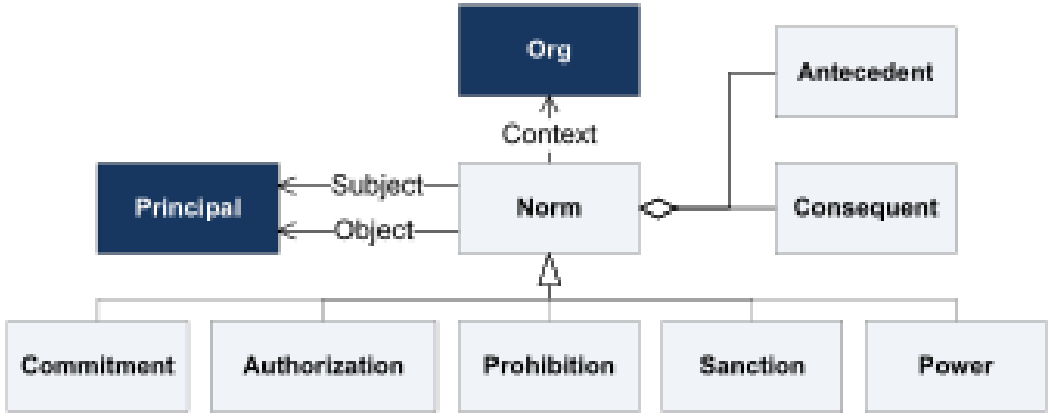
- Further details can be found in protocol guide:

  http://docs.jboss.org/scribble/releases/2.0.x/protocolguide/html/

Research – policies

- Commitments
  - Commitments enable an understanding of the obligations undertaken by different parties in a collaboration, which can be discharged under certain conditions.
  - For example, could be used to understand the net trading positions between different counterparties within a financial trading organisation.
  - Papers: http://www.csc.ncsu.edu/faculty/mpsingh/papers

Research – norms (1)



"Norms as a Basis for Governing Sociotechnical Systems"

Munindar P. Singh and Kartik Tadanki
North Carolina State University
June 16, 2010

Research – norms (2)

- Commitment

  “An active commitment means the subject (i.e. debtor) is committed to the object (i.e. creditor) within the scope of the organizational context [Singh et al., 2009]. It means that if the antecedent holds, the debtor commits to bringing about the consequent. And when the consequent holds, the commitment is satisfied and deactivated. Example: A researcher who borrows an instrument for a study commits to returning it within one hour of being requested to do so.”

- Authorization

  “Example: An instrument owner authorizes a colleague to use the instrument between 7:00PM and 9:00PM.”

- Prohibition

  “Examples: An instrument owner prohibits a borrower from changing the firmware on the instrument. A dataset curator prohibits a reader from publishing any of the data on an external web site.”

Research – norms (3)

- Sanction

  “Examples: An instrument owner would sanction a borrower who illicitly changes the firmware on a borrowed instrument by giving the borrower a poor rating. A dataset curator would sanction a reader who publishes any of the data externally by complaining to the Org. The resource sharing Org would sanction a reader who publishes any of the data externally by ejecting him from the Org.”

- Power

  “Examples: The Chesapeake Bay Org is empowered to admit or eject its members by declaring so. An instrument owner is empowered to contribute her instrument to a resource sharing Org, also by declaring so. A system administrator is empowered to admit new people into OOI by creating their accounts, but is—crucially—prohibited from creating accounts (and admitting members) without approval from the membership department. However, because the administrator has the power, her creation of a new account will succeed, though it might later be deemed illicit and revoked, and the administrator sanctioned for exercising the power illicitly.”

Research – capacity planning

- A choreography gives us a model of interactional behaviour across distributed services.
- A physical model of the distribution of services and their resources can offer an understanding of the potential bottlenecks, and elasticity requirements, of a production environment required to deliver a certain level of performance.
- Such a model could be used to estimate the cost of running an architecture under different loads, in various production environments (public/private/hybrid clouds)

Research – activity monitoring

- Requirement for activity information to capture runtime behaviour of a system

- Potential uses:
  - Defect detection, across and within services
  - Performance profiling
  - Business activity monitoring/transaction analysis
  - Autonomic infrastructure management
  - Others ....